# ACE: Classification for Information Lifecycle Management

Gauri Shah, Kaladhar Voruganti
*IBM Almaden Research Center*
*650 Harry Road*
*San Jose, CA 95120*
*gauris, kaladhar@us.ibm.com*

Piyush Shivam[*]
*Dept. of Computer Science*
*Duke University*
*Durham, NC 27708*
*shivam@cs.duke.edu*

Maria del Mar Alvarez Rohena[*]
*Dept. of Computer Science*
*University of California*
*Santa Barbara, CA 93106*
*malvarez@cs.ucsb.edu*

## Abstract

One of the principal problems in Information Lifecycle Management (ILM) is to align the business value of data with the most cost-effective and appropriate storage infrastructure. In this paper, we present ACE: a framework of tools for ILM, that classifies data and storage resources, and generates a data placement plan for informed utilization of the available storage resources in the system. The goal of ACE is to design a data placement plan that provides cost benefits to an organization while allowing efficient access to all important data. To achieve this goal, ACE uses a policy-based approach to classify data and storage based on the metadata attributes and hardware capabilities respectively. The main advantage of using ACE is that it enables appropriate usage of under-utilized storage systems without extensive human intervention. Another key characteristic of ACE is that it uses a policy-based architecture to automate the process of data valuation and storage classification.

## 1. Introduction

Storage needs of most enterprises are increasing at an exponential pace because organizations are automating more and more of their business processes, which is in turn leading to more data getting digitized and stored persistently. A study conducted by IBM predicted that the 3.2 million exabytes of information that existed on earth in the year 2001 would reach 43 million exabytes by the year 2005 [7]. This large corpus of data poses many challenges for intelligent data and storage management.

Organizations can only continue to increase the size of their storage facilities or find intelligent ways to reduce the overall size and number of stored files. IDC reports that the amount of new storage capacity installed each year is increasing by almost 80% annually [5]. However, the percentage of useful data residing on these expensive storage systems is becoming a very small percentage of the overall storage space being utilized. SNIA [10] uses the term *Information Lifecycle Management (ILM)* to addresses this and other data management-related issues. Content management, hierarchical storage management (HSM), and storage resource management, e.g., [2, 9, 11] are some of the tools that are trying to solve this problem under the umbrella of ILM.

Most legacy data systems treat all data in the same manner, irrespective of the value of the data to the business. Transforming legacy storage infrastructures into business-value-aware infrastructures is a time-consuming and difficult process due to the following reasons: (1) distributed data stores, (2) lack of application-data relationships, (3) temporal nature of business value, and (4) non-triviality of data valuation.

In this paper, we present ACE, a new framework of tools for ILM, that transforms under-utilized legacy storage systems into ones where the right type of data resides on the right type of storage at the right time. The goal of ACE is to help system administrators classify an organization's data and storage resources appropriately so that the data is placed on a matching class of storage resources according to its business value. The key features of ACE are as follows:

**Provides Classification and Data Placement:** ACE semi-automates the processes of determining the business value of the data, identifying different classes of data based on their business values, identifying the different tiers of storage quality, and aligning the data classes with the right storage tier.

**Uses Policy-driven Business Valuation:** In order to aid administrators to specify the business value of data, ACE has a policy-driven valuation mechanism. The policies determine how the data gets mapped to different business values, and how the storage gets mapped to different tiers of

[*]This work was done when the author was a summer intern at the IBM Almaden Research Center.

storage quality. This policy specification is very flexible and customizable.

**Handles Temporal Business Value:** ACE's data classification and data placement engines are capable of handling the *temporal* nature of the data business values by monitoring the system and the changing metadata characteristics.

**Optimizes Performance:** ACE uses several novel algorithms that improve the classification performance by reducing the domain space of the data that needs to be processed as well as the policies used for classification. Due to space constraints, we omit the performance optimizations from this paper.

In the rest of this paper, we focus on the overall architecture and mechanics of the ACE framework. Detailed implementation and performance analysis are subjects of future work.

## 2. ACE Architecture

ACE provides a modular framework to scan an existing system for its data and storage resources, and gives a data placement solution to best utilize the available storage resources. The objectives of ACE are three-fold: (1) To provide a *business valuation* to the data based on mining of its metadata attributes. (2) To determine the different tiers of storage quality available based on the hardware capabilities of the storage resources. (3) To provide a suitable *data placement* to ensure informed use of the storage resources.

Figure 1 shows the high-level ACE architecture.

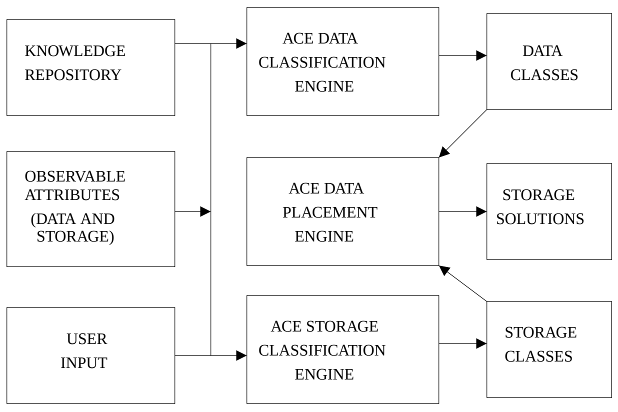
KNOWLEDGE REPOSITORY
OBSERVABLE ATTRIBUTES (DATA AND STORAGE)
USER INPUT
ACE DATA CLASSIFICATION ENGINE
ACE DATA PLACEMENT ENGINE
ACE STORAGE CLASSIFICATION ENGINE
DATA CLASSES
STORAGE SOLUTIONS
STORAGE CLASSES

**Figure 1. ACE architecture**

In order to achieve its objectives, the ACE framework has the following three main components: Data Classification Engine, Storage Classification Engine, and Data Placement Engine.

**Data Classification Engine:** This component is responsible for mining the metadata attributes of the data, and providing an appropriate business value to the data based on the policies. Different kinds of policies are used as explained in Section 3.2. The output of this engine is a set of data classes each of which comprises a collection of data objects which have similar characteristics and the same business value. Thus, there is a 1:1 mapping between the valuation provided to the data and the data classes generated using these values. We can use the range of $[1-9]$ to assign business values to the data; 9 represents mission-critical data, and 1 represents orphan/duplicate data that can be deleted.

**Storage Classification Engine:** This component is responsible for mining the capabilities of the available storage subsystems, and classifying them into different tiers of storage quality based on their ability to provide support for objectives such as disaster recovery, performance, availability, etc. The output of this engine is a set of storage classes each of which provides similar storage characteristics for the data that is stored in it.

**Data Placement Engine:** This component glues the two classification engines together. It uses the data classes and the storage classes to determine a storage plan for data placement. It recommends which data needs to be placed on which storage resource in order to meet the business needs of the application that uses the data. Thus, it suggests a storage plan that allows for informed use of the storage resources available in the system, and yet allows the data to be accessed efficiently as required by applications.

We note that ACE can easily be incorporated into products such as IBM TSM [11] and IBM San File System [8]. Both these products allow seamless transfer of data between different underlying storage devices while maintaining a uniform namespace for the user. In addition, the data classification engine can be used in other HSM migration systems such as given in [3] to determine candidates suitable for HSM.

## 3. Internal Details

In this section, we cover the details of each individual component of the ACE framework.

### 3.1. Input

The three sources of input to ACE as shown in Figure 1 are as follows:

**Knowledge Repository:** This repository is a collection of *policies* that encapsulate domain knowledge for data (or storage) classification. Each policy consists of a set of observable attributes of the data (or storage), the corresponding attribute values, and a business value for the data (or storage) that matches these attribute values. Each data/storage object is compared with all the policies to determine which one suits it best.

**Observable Attributes:** This input comes from mining the attributes of the data objects and the capabilities of the storage resources themselves. For example, we can monitor different attributes of the data such as file type, size, last update time, etc. These attributes are used in conjunction with the policies in the knowledge repository to classify the data into buckets of different business values, and the storage into different tiers of quality.

**User Input:** The administrator can provide additional input and hints about how to classify the data. For example, she can give sample data objects with business values. She can also create new customized policies that can be added to the knowledge repository. Finally, she can override the suggestions from ACE, and modify the data and storage classes or the data placement solution.

## 3.2. Classification Mechanics

Classification in ACE is done using a simple grouping of all data objects with the same business value into a single class. Business values are assigned to data objects using policies. The classification engines mine the attributes of the data or the storage subsystem, and then check to see which of the policies can be used to classify the objects. In ACE, there are three types of policies which are used to classify the data and the storage as follows:

**Knowledge-based policies:** Knowledge-based policies come pre-packaged with the ACE framework. This information is collected over a period of time in consultation with experts, and is based on experience. ACE uses a different set of policies based on the domain of the data that is being classified. For example, data for a medical application like X-rays files on a server is classified using different knowledge policies than the documents that are used for personal purposes.

**Expert-based policies:** Expert-based policies allow the administrator to rank attributes relatively to formulate a policy. The administrator chooses a set of attributes and assigns relative ranks to them. She also gives scores to different attributes values[1]. These scores are then combined and normalized to form a policy function. An object's metadata attributes are given as input into this policy function, and the normalized output value gives the business value of the object. Expert-based policies are flexible but they require advanced knowledge to be formulated on the fly.

**Example-based policies:** Instead of giving an expert policy, the administrator can give a sample set of files along with their business values. ACE will then mine the attribute values of the sample data set, and the associated business values to come up with a policy that meets the specified business value. The example files serve as a training set to standard machine learning techniques such as regression and decision trees to learn a policy function for classification.

[1]These values can also be ranges for certain attributes such as access time.

## 3.3. Data Classification Policies

Data classification is done by mining the metadata attributes of the data objects. Here, we consider that the data object is a *file*. Some of the metadata attributes that are used for data classification in ACE are: owner, access rights, application usage, file size, file type, last read time, last write time, create time, extension, access frequency, growth of file, number of applications using a file, etc.

ACE obtains metadata attributes either by scanning the file system or by parsing trace files of the file system. Table 1 shows some sample data classification policies; business values range from $[1-9]$. We note that although we talk about mining metadata attributes of objects to classify them, our policy-based approach is extensible to also incorporate content-based attributes [1, 4, 6].

## 3.4. Storage Classification Policies

The term *storage classification* refers to the classification of not just the storage devices, but it also includes the classification of the entire path from the host HBA ports to the storage device via network fabric. We initially begin with classification at the storage controller level. Some of the attributes used for storage classification in ACE are: random I/O capability, capacity, utilized capacity, max throughput, current throughput, max IOPs, observed IOPs, checksum available, encryption capability, access authentication, supported protocols, WORM capability, replication support, physical dimensions, power consumption, active/active capability, firmware swapping, multipathing software, MTTR, MTBF, cost, disk RPM, etc. Table 2 shows some sample storage classification policies. Storage classes range from $[1-9]$ with 9 being the highest quality class. Note that the storage is classified into different classes based on the objective such as disaster recovery, performance, etc. i.e. different attributes are considered for classification based on the objective.

## 3.5. Data Placement

Data placement involves matching the data classes to the appropriate storage classes. We want to store the most important data on the best-quality storage, and store the least important data in the lowest quality storage. In general, ideal data placement is a challenging problem. We take a simple approach of matching the highest data class with the

| Policy Name | Business Value | Attribute 1 | Attribute 2 | Attribute 3 |
|---|---|---|---|---|
| Rarely Accessed Docs | 5 | ATIME $\in < 45, \infty >$ days | DIR=DOCUMENTS | EXT=.OFFICE |
| Frequently Accessed Docs | 9 | ATIME $\in < 0, 7 >$ days | DIR=DOCUMENTS | EXT=.OFFICE |
| Moderately Accessed Media | 7 | ATIME $\in < 8, 45 >$ days | EXT=.MEDIA | |

**Table 1. Table showing some sample data classification policies. CTIME = Creation time, ATIME = Last Access Time, EXT = Extension. Some values represent ranges e.g. $< 0, 90 >$ represents between $0$ and $90$ days. Some of the values such as .CODE and .OFFICE actually represent an array of values.**

| Policy Name | Storage Class | Attribute 1 | Attribute 2 | Attribute 3 |
|---|---|---|---|---|
| **Disaster Recovery** | | | | |
| DR Tier 1 | 9 | Continuous Copy=SYNC | % Utilized $\in < 0, 50 >$ | Active-active=YES |
| DR Tier 4 | 6 | Snapshot Copy=YES | Cost $\in < 0, 50 >$ | Random I/O=YES |
| **Performance** | | | | |
| Highest Performance | 9 | Avail. BW $\in < 75, 100 >$ | Cache $\in < 128, 256 >$ | Disk RPM=15 |
| Medium Performance | 7 | Avail. BW $\in < 25, 74 >$ | Cache $\in < 128, 256 >$ | Disk RPM=10 |

**Table 2. Table showing some sample storage classification policies for different objectives.**

highest available storage class that has free storage to store the new data. Similarly, we match the moderately important data with the medium quality storage classes, and the least important data with the lowest quality storage classes. If space is not available in a particular storage class, ACE recommends the next best available storage class (which may be better or worse than the ideal storage class), or it informs the storage administrator to add more storage in the recommended class.

## 4. Conclusions and Future Work

In this paper, we present ACE, a new architecture to perform data and storage classification, and data placement using metadata attributes of files and capabilities of storage resources. We have demonstrated that ACE is beneficial in identifying the business valuation of data and assigning data to the appropriate storage hardware for informed resource utilization. We believe that ACE is a first step in a comprehensive solution for Information Lifecycle Management and complements existing HSM products that consider mainly age to migrate data. Future work includes implementation, and a detailed analysis of the performance of the ACE framework along with the cost benefits that it provides to an enterprise. We are also interested in deploying ACE with a data movement system such as TSM [11]. Optimal data placement instead of naïve data-storage matching is also an interesting direction to pursue for future work.

## References

[1] I. S. Dhillon and D. S. Modha. Concept Decompositions for Large Sparse Text Data using Clustering. *Machine Learning*, 42(1):143–175, Jan. 2001.

[2] EMC Documentum. `http://www.documentum.com`.

[3] D. He, X. Zhang, and D. H. Du. Parallel Hierarchical Storage Management in Object-based Cluster File Systems. In *Proceedings of the Fourteenth NASA Goddard Conference on Mass Storage Systems and Technologies*, May 2006.

[4] D. Koller and M. Sahami. Hierarchically Classifying Documents Using Very Few Words. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 170–178, July 1997.

[5] J. T. McArthur. Storage Networking: Business Drivers for SANs. *IDC*, 2003.

[6] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Text Classification from Labeled and Unlabeled Documents using EM. *Machine Learning*, 39(2/3):103–134, 2000.

[7] B. Rudolph. Keynote Speaker at 2001 IBM Storage and Storage Networking Symposium.

[8] IBM SAN File System. `http://www.ibm.com/software/tivoli/products/totalstorage-sfs/`.

[9] SGI InfiniteStorage Data Migration Facility (DMF). A White Paper. `http://www.sgi.com/pdfs/3631.pdf`.

[10] Storage Networking Industry Association. `http://www.snia.org/home`.

[11] IBM Tivoli Storage Manager. `http://www.ibm.com/software/tivoli/products/storage-mgr/`.